

The SanDisk Brand

- A global leader in storage technology
- Most compelling value proposition
- Unmatched innovation and IP
- We are Data Champions

SanDisk®
a Western Digital brand



WD Western Digital®



a Western Digital brand



a Western Digital brand

The Better Alignment of Managed Flash To System Behavior

Alex Lemberg
SW Manager

Agenda

- Flash Storage in Mobile & Embedded
 - Real Performance Requirements
 - The Gap Between Synthetic and User Activities
- Usage Case – Performance Peaks
- How to Handle Performance Peaks in Flash Management Architecture
- How it Affects the Endurance
- Driver Support

How to Get Better Performance?

Embedded Flash Memory is Everywhere



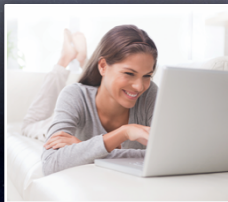
MOBILE



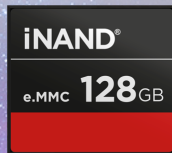
IoT



WEARABLES



COMPUTE



AUTO



INDUSTRIAL



HOME

The “Real” Storage Performance Requirements

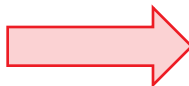
- What is the Most Important Performance Metric?

- Synthetic Benchmarks



Sequential Write (MB/Sec)
Sequential Read (MB/Sec)
Random Write (IOPS)
Random Read (IOPS)
SQL Insert/Update/Delete

- System Analysis



IO Latency
IO Flow
IO Stack Level

- User Experience



App Launch Time
Boot Time
Multitasking
Etc.



Getting IO Metrics – Is the Key

24-96 Hours of intensive “managed” user activity



Statistics

System Analysis &
Research

Simulations \
Testing

Wide platforms coverage

User Experience

Various Android &
Linux Versions

7 OEMs

16GB-64GB

1GB-4GB RAM

High End/Mid Range

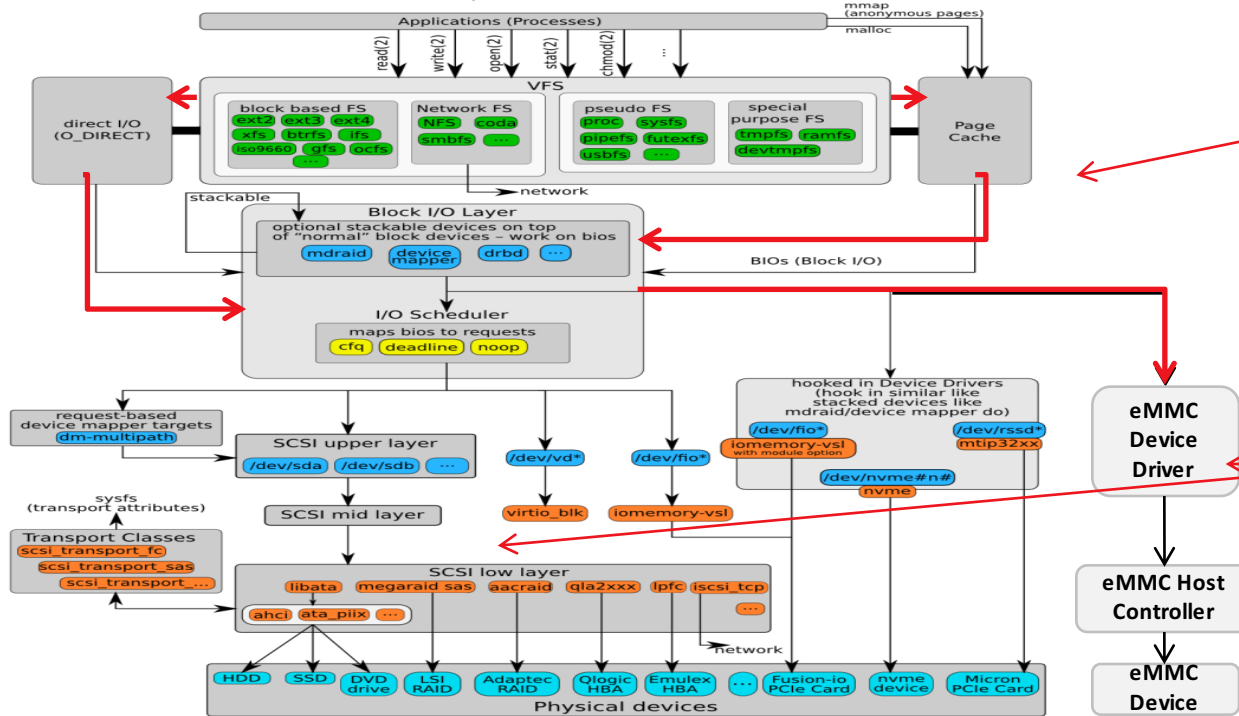
EXT4/F2FS

Regions

Enhanced Low Level Tracing

The Linux I/O Stack Diagram

version 1.0, 2012-06-20
outlines the Linux I/O stack as of Kernel version 3.3

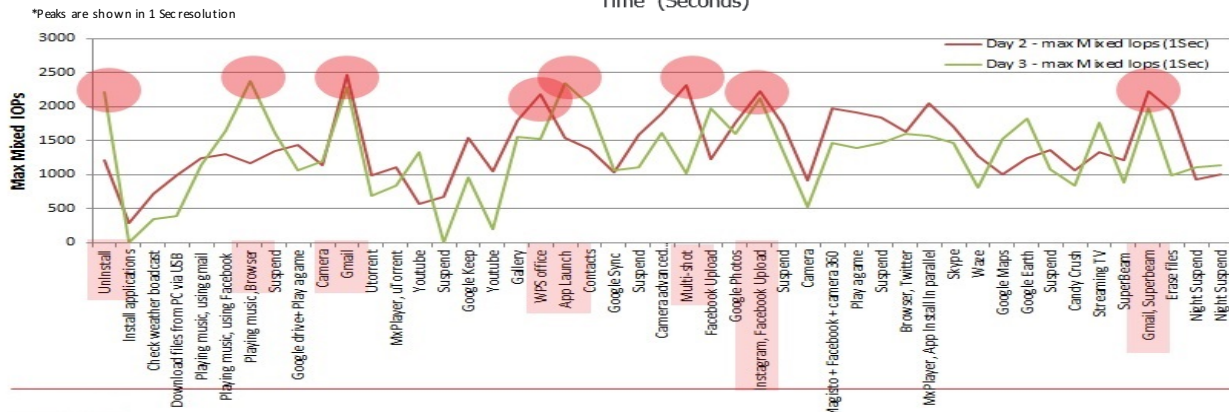
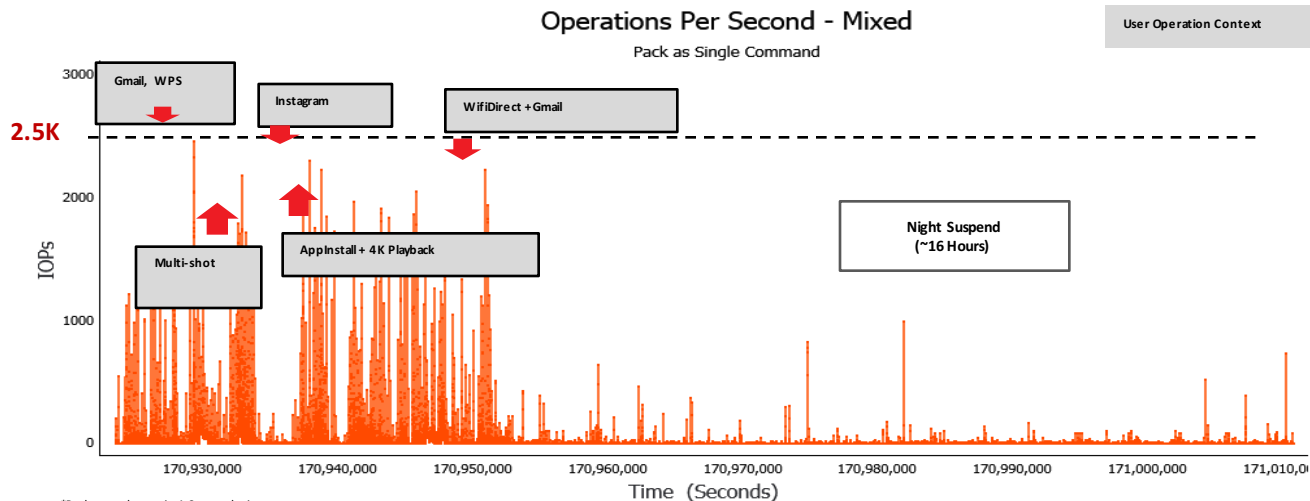


Catch the
Process and FS
Info

Trace point

The Linux I/O Stack Diagram (version 1.0, 2012-06-20)
<http://www.thomas-krenn.com/en/rss/linux-io-stack-diagram.html>
Created by Werner Fischer and Georg Schönberger
License: CC-BY-SA 3.0, see <http://creativecommons.org/licenses/by-sa/3.0/>

Enhanced Low Level Tracing Allows to Gather Per-Process Stat.

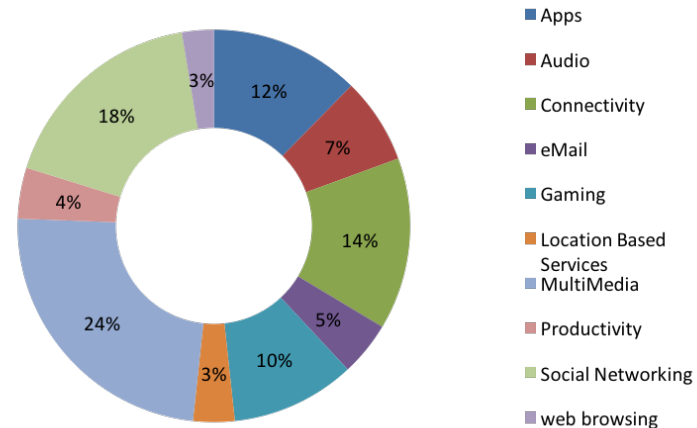


Agenda

- Flash Storage in Mobile & Embedded
 - Real Performance Requirements
- Usage Case
 - The Gap Between Synthetic and User Activities
 - Performance Peaks
- How to Handle Performance Peaks in Flash Management Architecture
- How it Affects the Endurance
- Driver Support

Usage Case – Description in High Level

High Level Category	Description	Duration (minutes)
Social networking	Facebook, twitter, Instagram	67
Multimedia	Camera + advanced features (multi-shot, photo editing), Video recording, Video on-line, Streaming, etc	91
eMail	Gmail	17
Connectivity	USB transfers, Cloud download & sync (Google Drive/drop box, Google Sync), wifi direct transfer	54
Audio	MP3 playback	27
Gaming	Minion Rush, Candy Crush	38
Apps	Playstore installs & search, Frequent Apps launch	47
Productivity	Google keep, WPS office	16
Location based services	Waze, Google Maps, Google Earth	13
Web browsing	Chrome, Firefox - Browser search, open URLs	7



Example of 2016 Flagship Mobile Phone - 32GB Storage

	Day1	Day2	Day3	Daily average
Total Research Time (Hours)	24	24	24	24
User Active Time (Hours)	8	8	8	8
User StandBy Time (Hours)	16	16	16	16
Storage Busy time (Min)	~15	~15.65	~19	~16.55
Write (GB)	12.57	11.22	12.86	12.21
Read (GB)	66.49	69.63	87.64	74.58
Discard (GB)	6.39	5.94	6.74	6.35
Flush#	103,136	211,371	228,335	180,947

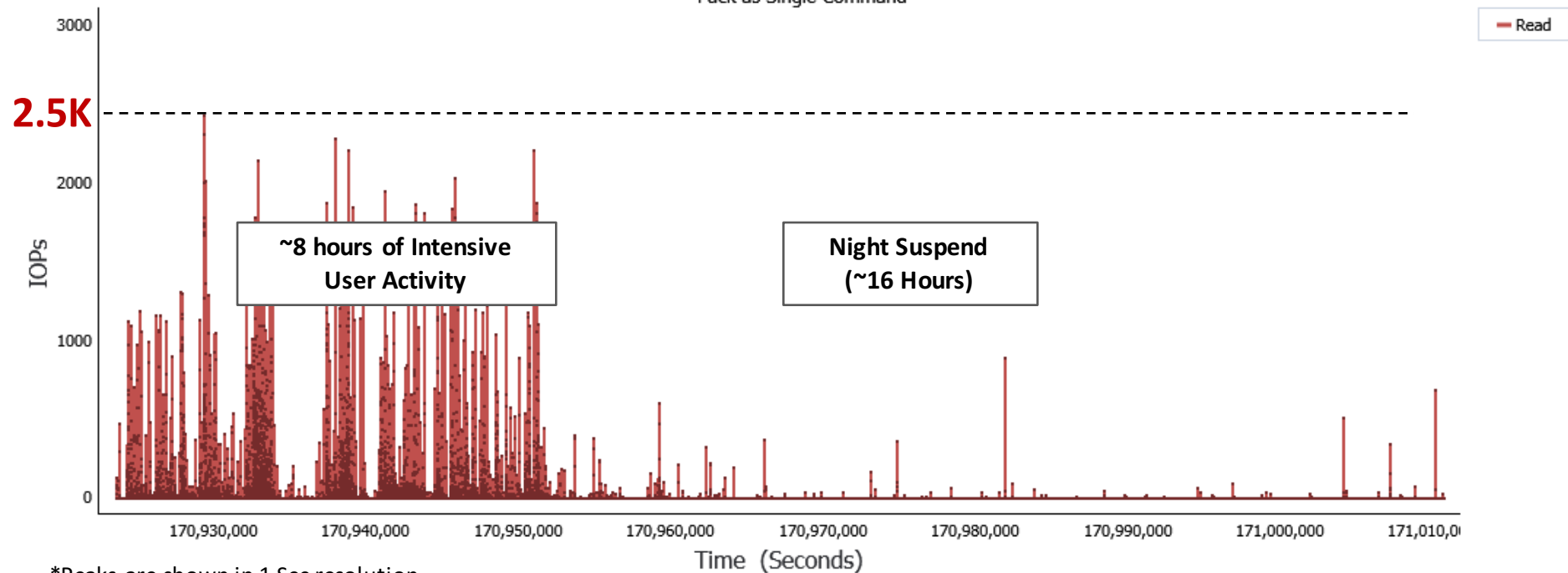
Only 15 Minutes Storage Busy Time (Out of 24 Hours)

Multi-Tasking Apps Are Not As Stressful As Synthetic Benchmark

<2.5K Read IOPs Peaks On Highly Intensive Daily Usage Case

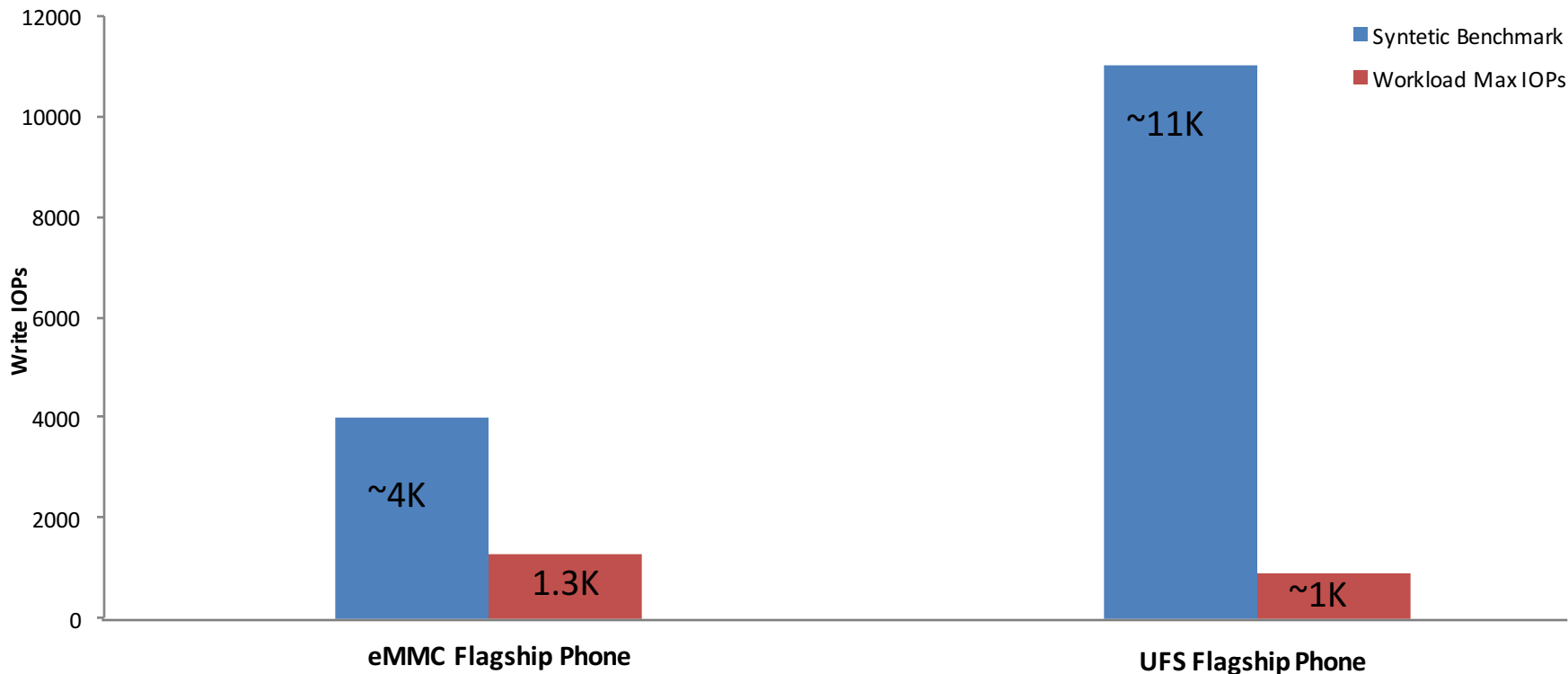
Operations Per Second - Read

Pack as Single Command



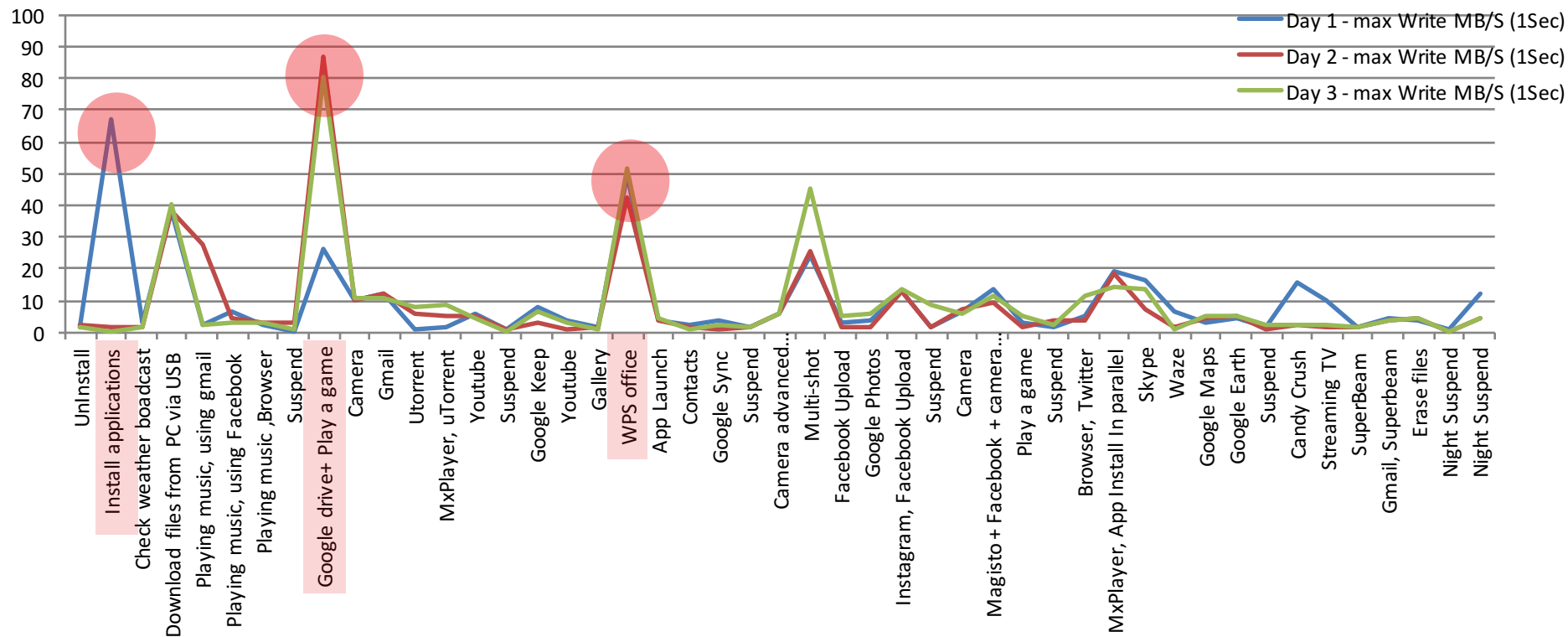
*Peaks are shown in 1 Sec resolution

The Gap Between Real Life And Synthetic Benchmarks



Max Write MB/S (Peaks)

Measured on Leading Android-Based Smartphone
Usage Case : 3 Days of intensive user activity



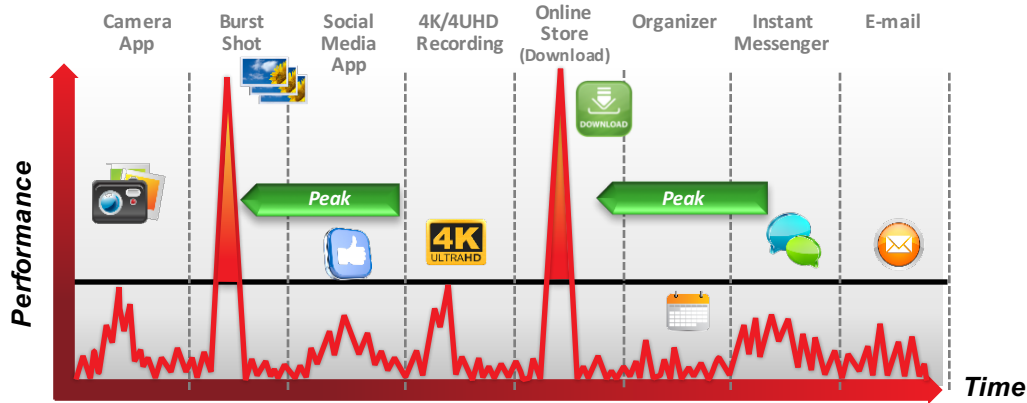
Agenda

- Flash Storage in Mobile & Embedded
 - Real Performance Requirements
- Usage Case
 - The Gap Between Synthetic and User Activities
 - Performance Peaks
- How to Handle Performance Peaks in Flash Management Architecture
- How it Affects the Endurance
- Driver Support

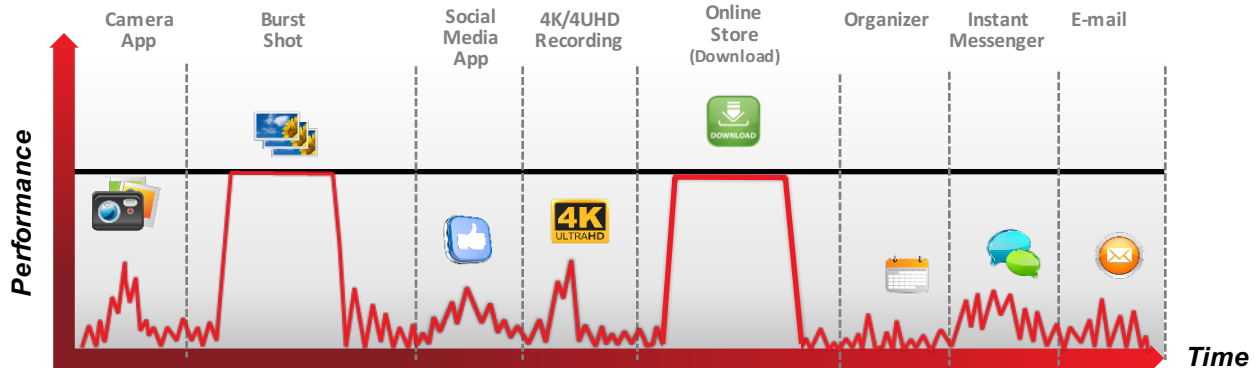
Handling Performance Peaks

Intelligent Peak Performance On-Demand

Peak-Awareness
Embedded
Storage



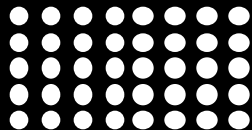
Typical
Embedded
Storage



Peak-Awareness Architecture

Application's Storage
Requests

Data

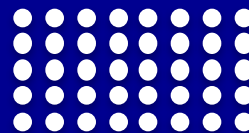


Storage Device Solution

Controller

FTL engine

SLC Buffer



Burst Storage

Main
Memory
Area

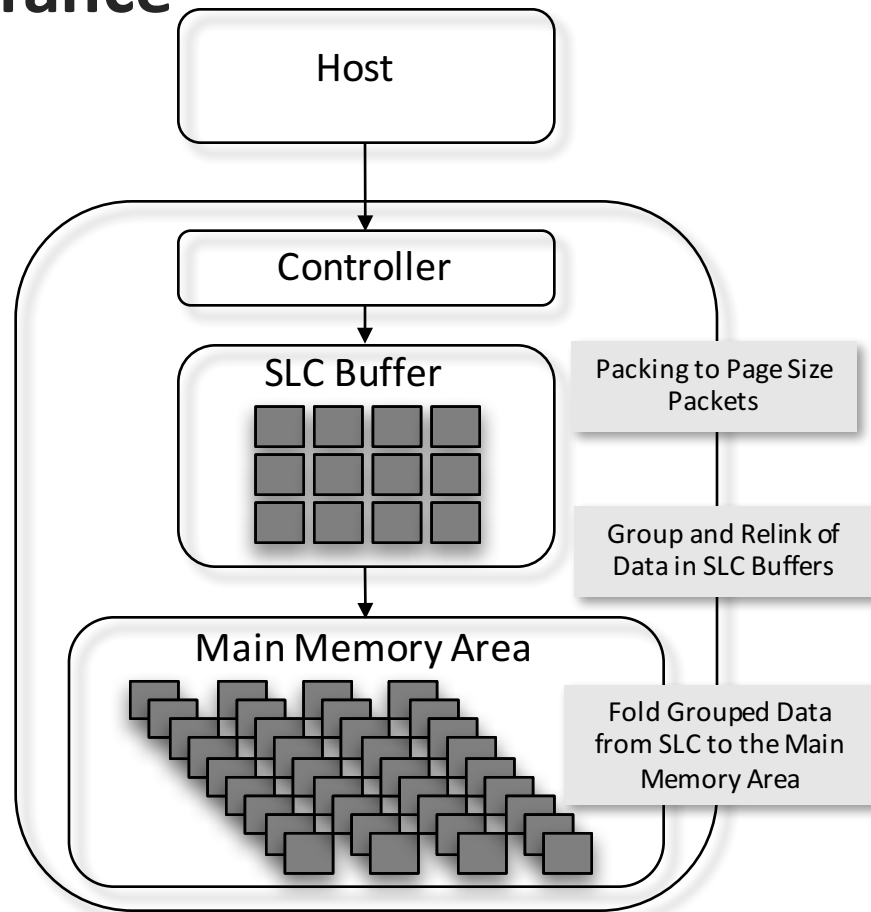
Agenda

- Flash Storage in Mobile & Embedded
 - Real Performance Requirements
- Usage Case
 - The Gap Between Synthetic and User Activities
 - Performance Peaks
- How to Handle Performance Peaks in Flash Management Architecture
- How it Affects the Endurance
- Driver Support

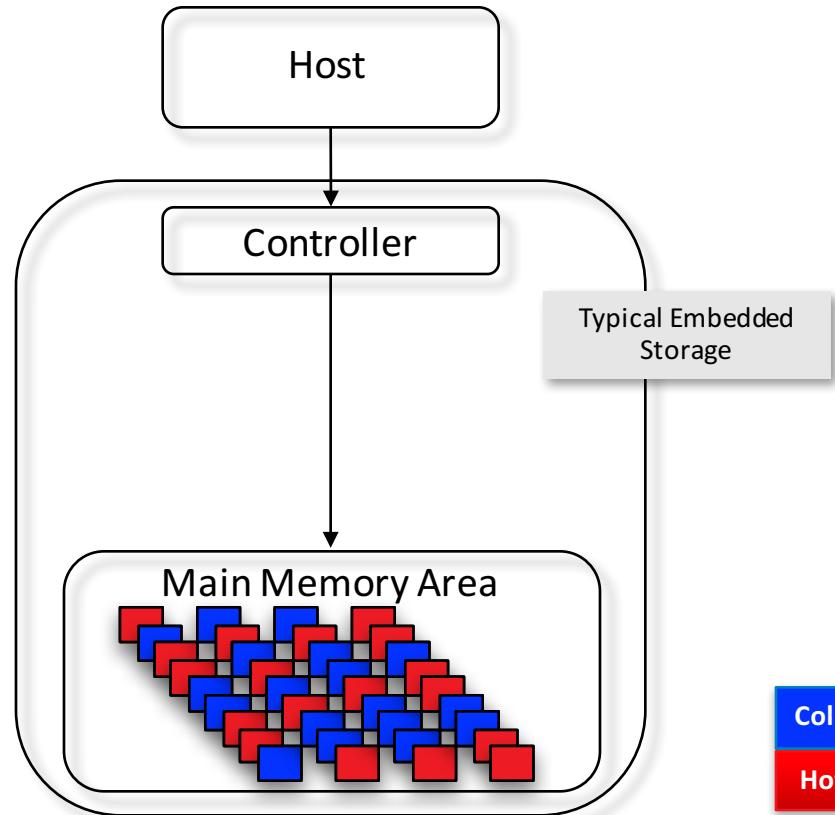
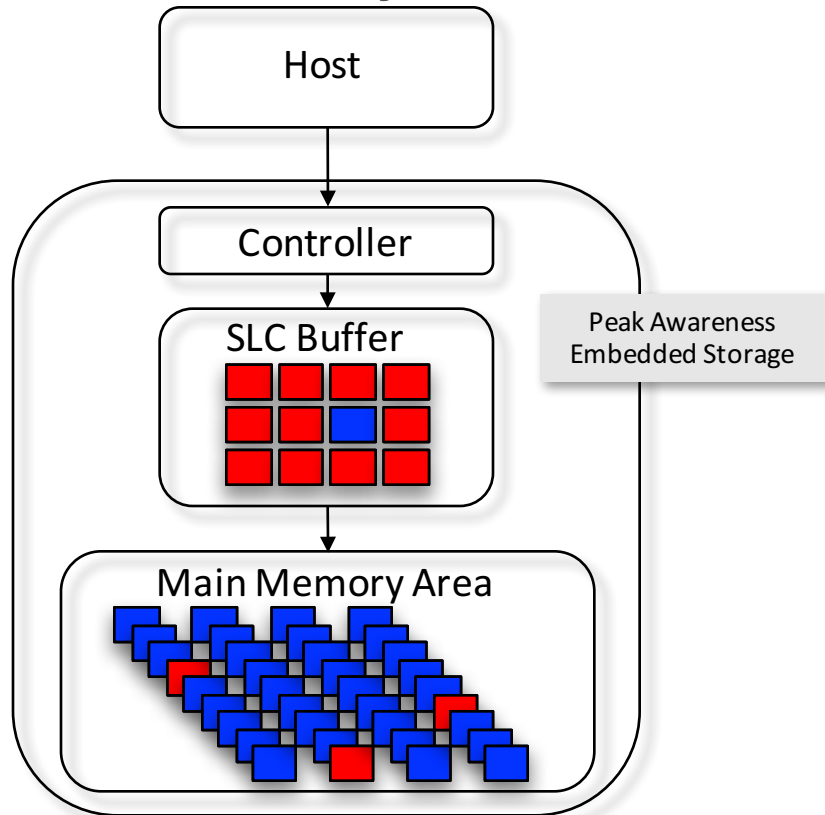
SLC Buffering is Good for Endurance

Extend product lifetime

- Host data is 'cached' in the SLC area
- The flash-management optimize data folding from SLC to the Main Memory Area, and by that minimize the write amplification
- Frequently accessed data ('Hot data') remains in SLC area and being update there
- Only 'Cold data' is folded and stored in main area by minimizing main-area program erase cycles



Observe Hot-data in SLC and Route Mostly Cold-data to the Main Memory Area



Cold-Data

Hot-Data

Agenda

- Flash Storage in Mobile & Embedded
 - Real Performance Requirements
- Usage Case
 - The Gap Between Synthetic and User Activities
 - Performance Peaks
- How to Handle Performance Peaks in Flash Management Architecture
- How it Affects the Endurance
- Driver Support

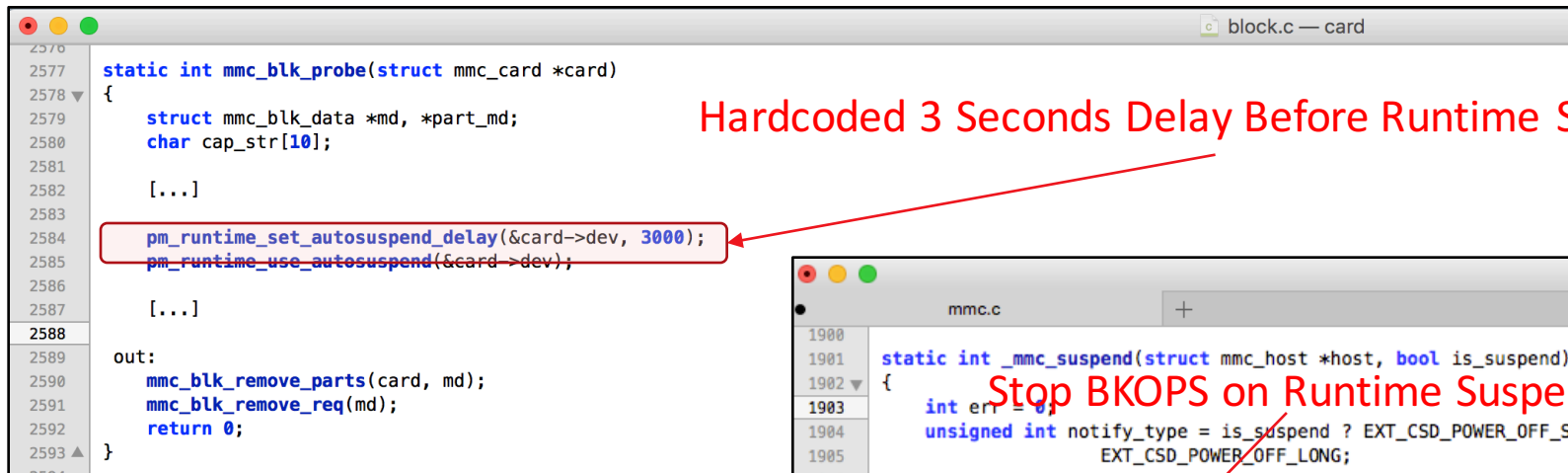
Is System Ready for Peak-Awareness Architecture?

- The Storage Driver's Power Management Flow is Not Always Adjusted to Peak-Awareness Architecture:
 - No time for Background Garbage Collection
 - Enter Sleep Mode Immediately on Suspend
 - User May Suffer from Hiccups after Resume
- Discard is Not Enabled on Some Systems
 - No Free Blocks for Internal Garbage Collection
 - Higher Write Amplification
 - Higher Latency

How to Adjust The Peak Awareness Flash Management

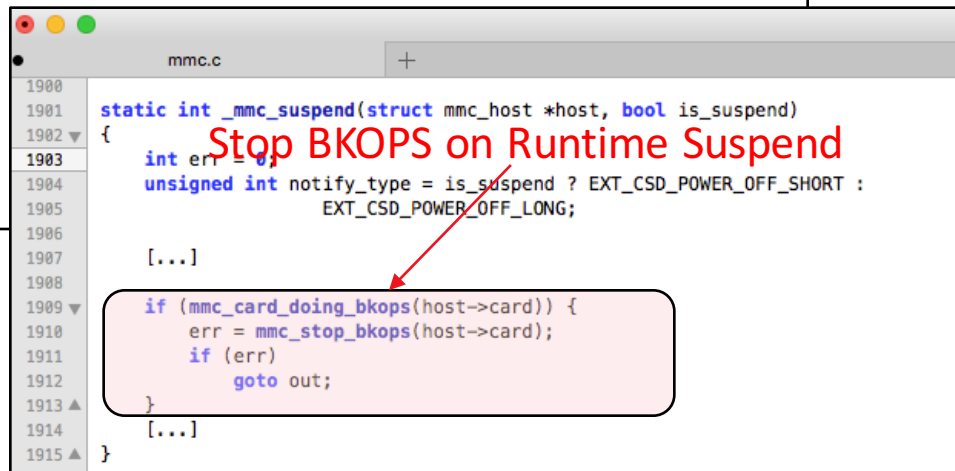
- Enable BKOPS Support
 - Both "Manual" and "Auto"
- Give Enough Time for BKOPS before Runtime Suspend
- Enable PowerOffNotification to Allow Background GC
 - Set PowerOffNotification ON on card init
- Enable DISCARD
 - FS Mount Flag
 - Fstrim on Android

The Problem - Need to Give Enough Time for BKOPS Before Runtime Suspend



```
2576
2577 static int mmc_blk_probe(struct mmc_card *card)
2578 {
2579     struct mmc_blk_data *md, *part_md;
2580     char cap_str[10];
2581
2582     [...]
2583
2584     pm_runtime_set_autosuspend_delay(&card->dev, 3000);
2585     pm_runtime_use_autosuspend(&card->dev);
2586
2587     [...]
2588
2589 out:
2590     mmc_blk_remove_parts(card, md);
2591     mmc_blk_remove_req(md);
2592     return 0;
2593 }
```

Hardcoded 3 Seconds Delay Before Runtime Suspend

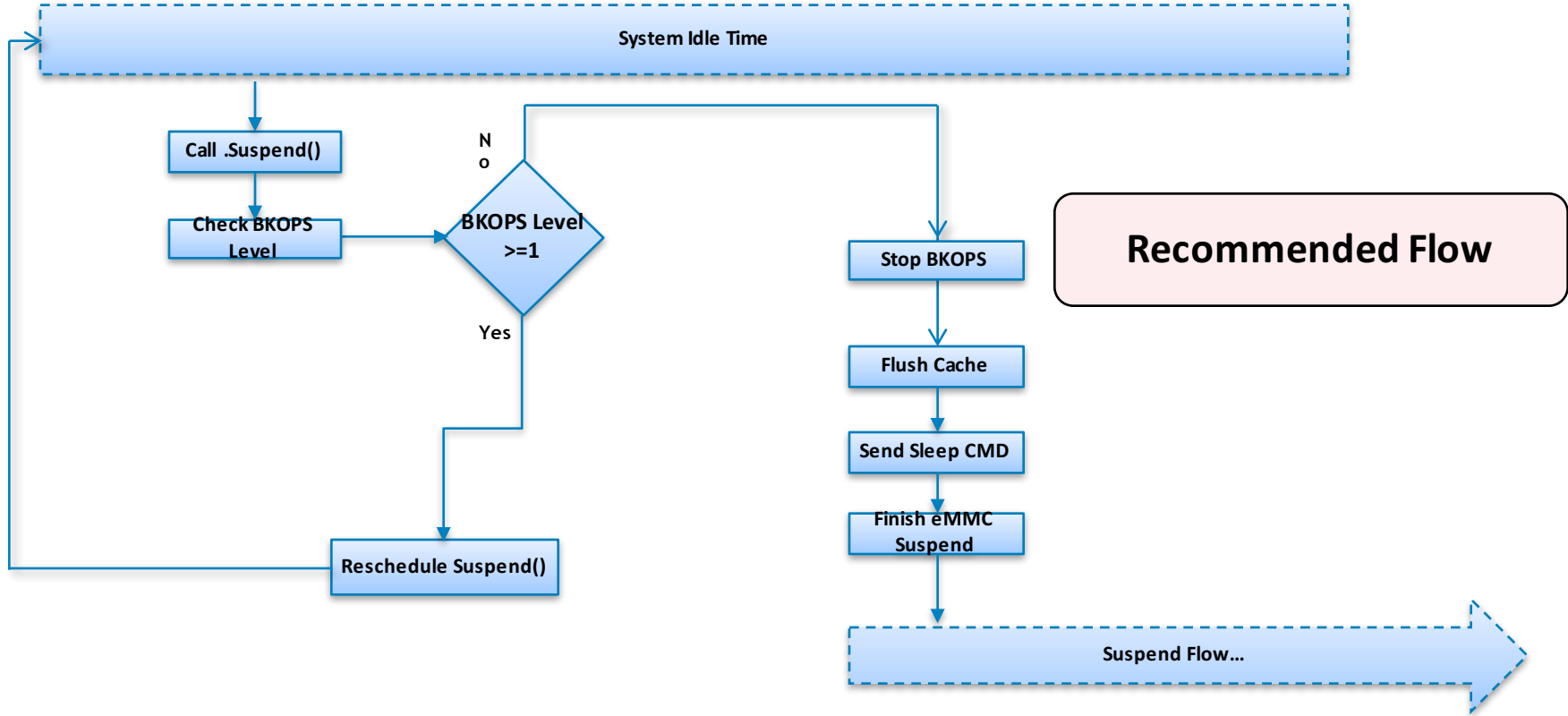


```
1900
1901 static int _mmc_suspend(struct mmc_host *host, bool is_suspend)
1902 {
1903     int err = 0;
1904     unsigned int notify_type = is_suspend ? EXT_CSD_POWER_OFF_SHORT :
1905                               EXT_CSD_POWER_OFF_LONG;
1906
1907     [...]
1908
1909     if (mmc_card_doing_bkops(host->card)) {
1910         err = mmc_stop_bkops(host->card);
1911         if (err)
1912             goto out;
1913     }
1914     [...]
1915 }
```

Stop BKOPS on Runtime Suspend

Check the BKOPS Status On Runtime Suspend

Runtime Suspend Flow with Auto BKOPS



Patchset for Handling BKOPS Status on Runtime Suspend

- Patchset Submitted to add BKOPS Status support in eMMC PM

- <http://marc.info/?l=linux-mmc&m=147274208821646&w=4>
- <http://marc.info/?l=linux-mmc&m=147274015121024&w=4>
- <http://marc.info/?l=linux-mmc&m=147274104021291&w=4>
- <http://marc.info/?l=linux-mmc&m=147274215821667&w=4>

```
18 diff --git a/drivers/mmc/core/mmc.c b/drivers/mmc/core/mmc.c
19 index e2e987f..c4c6326 100644
20 --- a/drivers/mmc/core/mmc.c
21 +++ b/drivers/mmc/core/mmc.c
22 @@ -1904,7 +1904,8 @@ static void mmc_detect(struct mmc_host *host)
23     }
24 }
25
26 -static int _mmc_suspend(struct mmc_host *host, bool is_suspend)
27 +static int _mmc_suspend(struct mmc_host *host, bool is_suspend,
28 +    bool is_runtime_pm)
29 {
30     int err = 0;
31     unsigned int notify_type = is_suspend ? EXT_CSD_POWER_OFF_SHORT :
32     @@ -1918,10 +1919,25 @@ static int _mmc_suspend(struct mmc_host *host, bool is_suspend)
33     if (mmc_card_suspended(host->card))
34         goto out;
35
36     if (mmc_card_doing_bkops(host->card)) {
37         err = mmc_stop_bkops(host->card);
38         if (err)
39             goto out;
40         if (mmc_card_doing_bkops(host->card) ||
41             host->card->ext_csd.auto_bkops_en) {
42             err = mmc_read_bkops_status(host->card);
43             if (err) {
44                 pr_err("%s: error %d reading BKOPS Status\n",
45                     mmc_hostname(host), err);
46                 goto out;
47             }
48             if (is_runtime_pm && host->card->ext_csd.raw_bkops_status >=
49                 EXT_CSD_BKOPS_LEVEL_1) {
50                 pm_schedule_suspend(&host->card->dev,
51                     MMC_RUNTIME_SUSPEND_DELAY_MS);
52                 goto out;
53             }
54             if (mmc_card_doing_bkops(host->card)) {
55                 err = mmc_stop_bkops(host->card);
56                 if (err)
57                     goto out;
58             }
59         }
60         err = mmc_flush_cache(host->card);
```

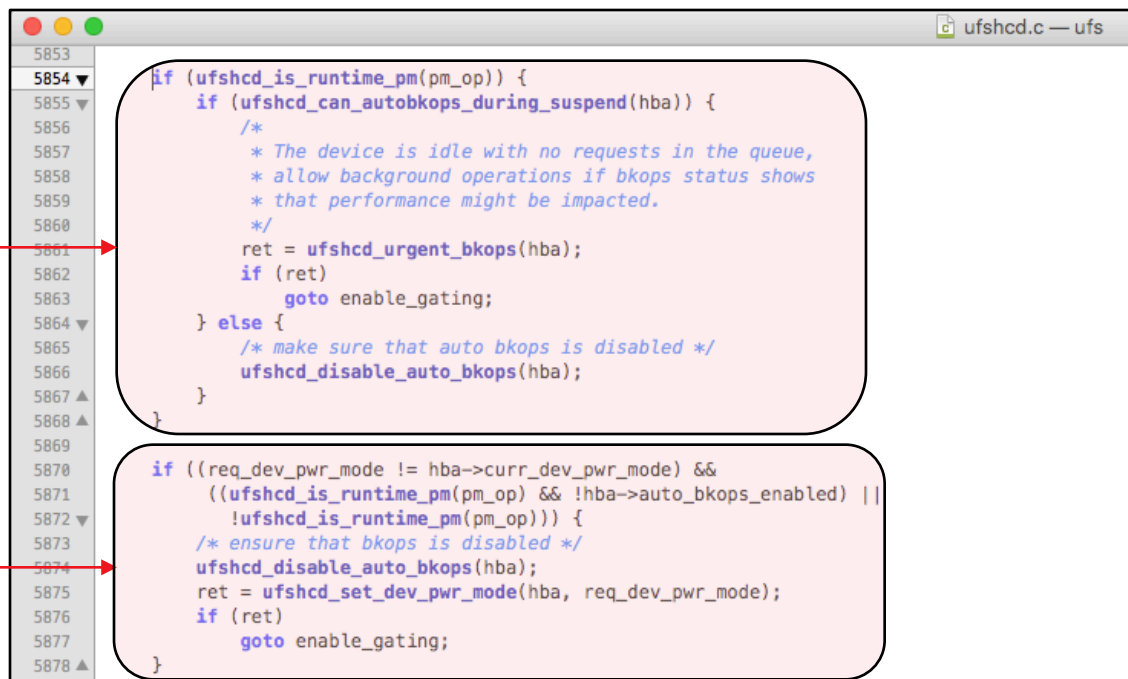
Check BKOPS Status & Reschedule Suspend

BKOPS in UFS

No Proper Handling of URGENT BKOPS in System Suspend

Allows URGENT BKOPS in Runtime Suspend

Disable BKOPS in System Suspend



```
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
```

```
if (ufshcd_is_runtime_pm(pm_op)) {
    if (ufshcd_can_autobkops_during_suspend(hba)) {
        /*
         * The device is idle with no requests in the queue,
         * allow background operations if bkops status shows
         * that performance might be impacted.
         */
        ret = ufshcd_urgent_bkops(hba);
        if (ret)
            goto enable_gating;
    } else {
        /* make sure that auto bkops is disabled */
        ufshcd_disable_auto_bkops(hba);
    }
}

if ((req_dev_pwr_mode != hba->curr_dev_pwr_mode) &&
    ((ufshcd_is_runtime_pm(pm_op) && !hba->auto_bkops_enabled) ||
     !ufshcd_is_runtime_pm(pm_op))) {
    /* ensure that bkops is disabled */
    ufshcd_disable_auto_bkops(hba);
    ret = ufshcd_set_dev_pwr_mode(hba, req_dev_pwr_mode);
    if (ret)
        goto enable_gating;
}
```

Q&A

Email: alex.lemberg@sandisk.com

Thank You!



SanDisk®, a Western Digital brand, is expanding the possibilities of storage. Our products are in the world's leading-edge data centers, advanced mobile devices and laptops, and trusted by consumers worldwide.

© 2016 Western Digital Corporation or its affiliates. All rights reserved.

SanDisk and iNAND are trademarks of Western Digital Corporation or its affiliates, registered in the United States and other countries. Android is a trademark of Google Inc. The microSD, microSDHC and microSDXC marks and logos are trademarks of SD-3C, LLC. Other brand names mentioned herein are for identification purposes only and may be the trademark(s) of their respective holder(s).

Western Digital Technologies, Inc. is the seller of record and licensee in the Americas of SanDisk® products.